Universität Innsbruck - Institut für Informatik
Prof. Clemens Ballarin, Robert Binna, Friedrich Neurauter, Francois Scharffe and Sarah Winkler

24 June 2008

Proseminar Algorithmen und Datenstrukturen

Exercise Sheet 13

Exercise 1 (Knapsack problem)

Consider the knapsack problem as explained in the lecture: Given $K \in \mathbb{N}$ the capacity of a knapsack and n objects $1, ..., n$. Each object has a weight of size $k_i \in \mathbb{N}$. Determine a subset of these objects that the sum of the weights of this subset is exactly the same as the capacity of the knapsack.

a) Implement an algorithm in C that solves the knapsack problem as described above.
Solution: See the file *knapsack.c*.

b) In the US, the president is elected by so-called electors which are representatives of a state citizens (check the following URL for more details: `http://en.wikipedia.org/wiki/U.S._Electoral_College`). The electors are nominated according to the votes of the citizens in a winner takes it all fashion. By this reason all electors of one state are for one candidate or the other one. So each state has a weight in the presidential elections equal to the number of electors. The question is: Can a draw occur as an outcome of the voting process? Give the formalization of this problem in terms of the Knapsack problem and use it to answer.
Solution:
The problems is formalized like the following: We consider S states in the US, each of them having e_s electors. In order to find out if it is possible to have the same number of electors voting for a candidate, we need to check if there is an exact solution to the Knapsack problem with the sack size equals to $\frac{\sum_{s \in S} e_s}{2}$. If there is a solution D, there will necessary be a set of states $S \setminus D$ such that $\sum_{s \in D} e_s = \sum_{s \in S \setminus D} e_s = \frac{\sum_{s \in S} e_s}{2}$. It is actually possible to get a draw.

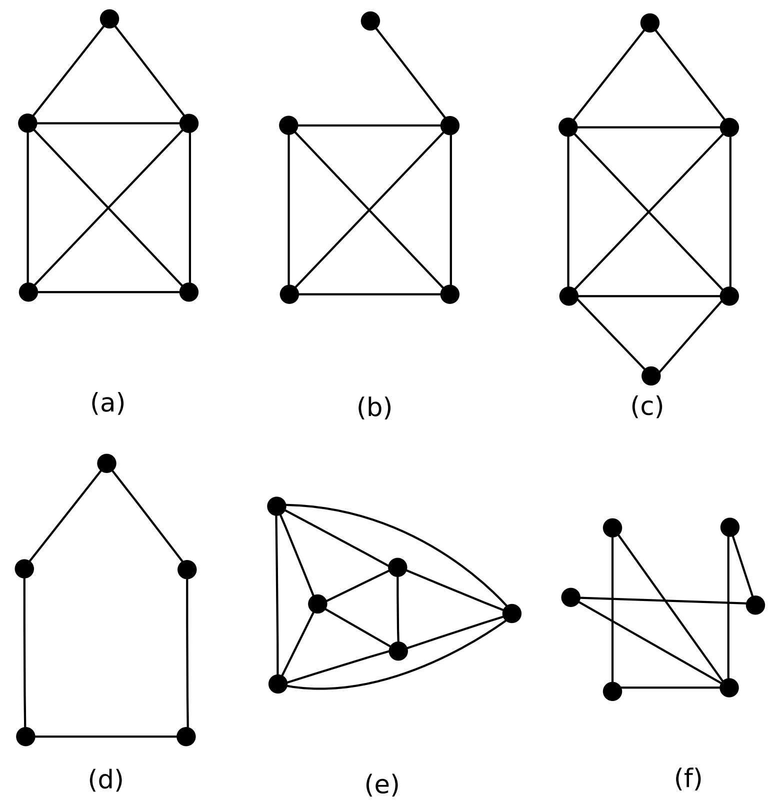

Abbildung 1: Are these graphs Eulerians ?

Exercise 2 (Graph algorithms)

Give three examples of problems that can be represented using graphs, and formalize them in terms of graphs. The model of graphs presented in the lecture may be extended by attaching information to nodes and edges, if necessary.

Exercise 3 (Euler Graphs)

Consider the graphs Figure 1. For each of them:

- Is there an Eulerian path ?
- Is there an Eulerian circuit ?

Solution:

- A connected graph has an Eulerian path iff it has at most two graph vertices of odd degree.

- A connected graph has an Eulerian circuit iff it has no graph vertices of odd degree.

Graph	Path	Circuit
a	Yes	No
b	No	No
c	Yes	Yes
d	Yes	Yes
e	Yes	Yes
f	Yes	Yes

Exercise 4 (Knapsack problem variant)

Solve the following variant of the knapsack problem: given n items $1...n$ where item i is of size k_i and a knapsack of size K. Determine a subset of items such that the sum of their sizes is maximal but less than or equal K.

a) Design a dynamic programming algorithm to solve the problem in $O(nK)$. Outline your algorithm and show that the computed solution is optimal.

b) Give pseudo code for your algorithm.

Solution:

Listing 1 Modified Knapsack

Input: $s[1 \ldots n]$ Array of elements, k: element
Output: *solution*: Array of elements (representing the maximal set of items fitting in the knapsack);

```
begin
  /* knapsack output a two dimensional array containing the solution */
  $P := knapsack(s, k)$;
  $j := k$;
  $m := 0$;
  while $!P[m][j].ex$ do
    $j := j - 1$;
  while $j > 0$ do
    $i := n$;
    while $!P[i][j].belong$ do
      $i := i - 1$;
    $solution[m] := P[i][j]$;
    $m := m + 1$;
    $j := j - s[i]$;
  return $solution$;
end
```